Multiple-Precision Correctly Rounded Gauss-Legendre Quadrature

Laurent Fousse

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Multiple-Precision Correctly Rounded Gauss-Legendre Quadrature

Laurent Fousse

N° 5705

Septembre 2005

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--5705--FR+ENG

Multiple-Precision Correctly Rounded Gauss-Legendre Quadrature

Laurent Fousse

Thème SYM — Systèmes symboliques
Projet Spaces

Rapport de recherche n° 5705 — Septembre 2005 — 17 pages

Abstract: Numerical integration is an operation that is frequently available in multiple precision numerical software packages. The different quadrature schemes used are considered well studied but the roundoff errors that result from the computation are often neglected, and the actual accuracy of the results are therefore seldom rigorously proven.

We focus on the Gauss-Legendre quadrature scheme and describe the algorithms needed in our implementation. A thorough error analysis is given as well as experimental error measurements and timings.

Key-words: numerical integration, correct rounding, Gauss-Legendre, multiple precision.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Intégration de Gauss-Legendre en précision arbitraire avec arrondi correct

Résumé : L'intégration numérique est une opération fréquemment disponible dans les logiciels de calcul numérique en précision arbitraire. Bien que les différentes méthodes d'intégration aient été bien étudiées du point de vue mathématique, l'erreur qui résulte des arrondis lors de l'évaluation est souvent négligée, et la précision finale des calculs n'est pas prouvée.

Nous étudions la méthode d'intégration de Gauss-Legendre et décrivons les algorithmes utilisés dans notre implémentation. Une analyse rigoureuse de l'erreur ainsi que des résultats expérimentaux sont fournis.

Mots-clés : intégration numérique, arrondi correct, Gauss-Legendre, précision arbitraire.

1 Introduction

Numerical integration is readily available in most multiple precision numerical computation software (e.g. Pari/GP, MuPAD, Mathematica, Maple, ...). The working precision can usually be changed between computations and the software displays the computed result with desired precision (some software cheat and use a greater internal precision than what is displayed). At this point the user is at a loss to determine how many if any of the digits in the returned result are correct. Let us try a simple example and ask MuPAD 2.5.3, Pari/GP 2.1.6, and Maple 9 for the value of $I = \int_{10^6}^{10^6+\pi} \sin(\sin(x))dx$ with 19 digits of precision:

```
>> DIGITS := 19: numeric::quadrature(sin(sin(x)), x=10^6.. 10^6+PI);
                          1.661291708545107576
```

```
? default(realprecision,19); intnum(x=10^6,10^6+Pi,sin(sin(x)))
%1 = 1.661291708545990308
```

```
> Digits:=19: evalf(Int(sin(sin(x)), x = 10^6.. 10^6 + Pi));
                       1.661291708545107059
```

So we get three different results depending on the software used, which is more than enough. This disappointing observation is easy to explain. Although the semantics of floating-point numbers computations is well defined when those computations are restricted to the four basic operations (thanks to the IEEE754 standard [1]), nothing is guaranteed as soon as computations are composed (think for example of the double rounding problem), or transcendental functions like $\sin$ are used. Hence the difficulty of a correctly rounded numerical quadrature operation.

Several approaches were made to surpass these shortcomings when computing integrals. One can mention the use of adaptive quadrature functions (in MuPAD [2]), or dynamic error control (of simple or multiple integrals [3, 4]). Our work differs from these approaches in that we seek a correctly rounded result for which a proven bound on the error is needed. We need to provide more than an estimation and instead give a guaranteed error bound on the final result which takes into account all errors, including the roundoff errors which are the toughest to measure properly. We emphasize the fact that computing the correct rounding of the integral is only possible if its value is not exactly representable by a floating-point number, in which special case the problem is undecidable and we can only provide a result with an absolute error bound as small as desired.

We first describe briefly the Gauss-Legendre integration from a mathematical point of view. A more detailed study of the Gauss family of integration methods with corresponding weight functions can be found in [5]. In this paper, $f : [a, b] \to \mathbb{R}$ is the C^∞ function we want to integrate on a finite domain $[a, b]$ and n is the number of points of the Gauss-Legendre method. Let

$$I = \int_a^b f(x)dx$$

be the exact value of the integral. We define the inner product of f and g on $[a,b]$ for the admissible weight function w as

$$< f, g >= \int_a^b w(x)f(x)g(x)dx.$$

This leads to the definition of a sequence of orthogonal polynomials $(p_i)_{i\geq 0}$ such that:

$$\forall i \in \mathbb{N}, \deg(p_i) = i$$

$$\forall (i,j) \in \mathbb{N}^2, < p_i, p_j >= 0 \quad \text{if } i = j$$

and the leading coefficient k_i of p_i is positive. For fixed $n > 0$, p_n has n distinct roots in $]a,b[$ which we name $x_0 < x_1 < \ldots < x_{n-1}$. The Gauss quadrature method associated to the weight function w on $[a,b]$ is the interpolatory method at evaluation points $(x_i)_{0\leq i<n}$ such that

$$\int_a^b w(x)p(x)dx = \sum_{i=0}^{n-1} w_i p(x_i)$$

holds for every polynomial p of degree at most $n-1$ (this is enough to define the weights w_i although the method will be shown to integrate accurately polynomials of degree at most $2n-1$).

The Gauss-Legendre quadrature method is the Gauss method for the weight function $w = 1$. Additionally the Legendre polynomials are usually defined on $[-1,1]$ and normalized such that $P_n(1) = 1$ and we will follow this custom here.

In section 2 we will describe the algorithms used to compute the Legendre polynomials, the evaluation points and the coefficients of the method. The mathematical error of the method will then be discussed. In section 3 the actual quadrature algorithm is explained and the error analyzed. We follow with experiments and some conclusive remarks.

2 Algorithms

In the rest of this paper P_n is the Legendre polynomials of degree n defined on $[-1,1]$ as usual. The quadrature method on $[a,b]$ is derived from the quadrature method on $[-1,1]$ from a shifting and scaling in the polynomial. If we name (V_n) the updated family of polynomials on $[a,b]$ we have the simple formulas

$$V_n(u) = P_n\left(\frac{2u-(b+a)}{b-a}\right) \quad \text{and} \quad P_n(x) = V_n\left(\frac{a+b+x(b-a)}{2}\right).$$

When translating the associated quantities (evaluation points and weights) from $[-1,1]$ to the target domain $[a,b]$ the details of the translation are omitted for the sake of simplicity. We will however take into account the error corresponding to this translation.

2.1 Legendre Polynomials

Like other orthogonal polynomial sequences, the $(P_n)_{n\geq 0}$ satisfy a recurrence relationship:

$$\left\{\begin{array}{rcl} P_0(X) & = & 1 \\ P_1(X) & = & X \\ (n+1)P_{n+1}(X) & = & (2n+1)XP_n(X) - nP_{n-1}(X). \end{array}\right. \quad (1)$$

From this we deduce that P_n has only monomials of degree the same parity as n and has rational coefficients. We assume that the common denominator is always a power of 2. Thus P_n can be written

$$P_n(X) = \left\{\begin{array}{cc} 2^{-d_n}Q_n(X^2) & \text{if } n \text{ is even} \\ 2^{-d_n}XQ_n(X^2) & \text{otherwise.} \end{array}\right.$$

The problem of computing P_n is reduced to that of computing Q_n and d_n. The procedure is detailed in Algorithm 1. Details for the case of the constant coefficient have been omitted on line 8.

Algorithm 1 Computation of the Legendre Polynomials

1: $Q_0 \leftarrow 1$
2: $Q_1 \leftarrow 1$
3: $d_0 \leftarrow 0$
4: $d_1 \leftarrow 0$
5: $p \leftarrow 0$ ▷ holds the parity of the polynomial currently computed
6: **for** $i \leftarrow 2$ to n **do**
7: $\quad \alpha \leftarrow d_{1-p} - d_p$
8: $\quad Q_p \leftarrow 2^{\alpha}(i-1)Q_p + (2i-1)X^{p-1}Q_{1-p}$
9: $\quad r \leftarrow \max(\{n \in \mathbb{N} \mid 2^n | i\})$
10: $\quad s \leftarrow \max(\{n \in \mathbb{N} \mid 2^n | Q_p\})$
11: $\quad Q_p \leftarrow \frac{2^{r-s}}{i}Q_p$
12: $\quad d_p \leftarrow d_{1-p} + r - s$
13: $\quad p \leftarrow 1 - p$
14: **end for**

2.2 Evaluation points

Computing the roots $(x_i)_{0\leq i<n}$ of P_n reduces to the computation of the roots of Q_n. Let $m = \lfloor \frac{n}{2} \rfloor$ and $u_0, u_1, \ldots u_{m-1}$ be the roots of Q_n, we have:

$$\{x_0, x_1, \ldots, x_{n-1}\} = \left\{\begin{array}{cc} \{\pm\sqrt{u_0}, \ldots, \pm\sqrt{u_{m-1}}\} & \text{if } n \text{ is even,} \\ \{\pm\sqrt{u_0}, \ldots, \pm\sqrt{u_{m-1}}, \} \cup \{0\} & \text{otherwise.} \end{array}\right.$$

where $0 < u_0 < u_1 < \ldots < u_{m-1}$. The process of computing the roots of Q_n has two steps:

1. root isolation, that is finding m intervals that contain each exactly one root of Q_n,
2. root refinement.

The root isolation is made using Uspensky's algorithm as described in [6]. The input of the algorithm is $Q_n(x)$, and the output is a sequence of $m+1$ intervals of the form $\frac{c_i}{2^{l_i}}$ where c_i and l_i are integers and such that $[\frac{c_i}{2^{l_i}}, \frac{c_i+1}{2^{l_i}}]$ contains exactly one root of Q_n, namely u_i. At this step, $\log_2(c_i)$ bits of u_i are known.

The refinement is done using dichotomy and Newton iteration.

2.2.1 Dichotomy

Performing a dichotomy to refine each root is a straightforward method but quite slow if done naively. Since $Q_n(X) = \sum_{i=0}^{m} a_i X^i$ has integer coefficients, it is possible to compute the sign of $Q_n(\frac{c}{2^l})$ as $\operatorname{sign}\left(2^{lm} Q_n(\frac{c}{2^l})\right) = \operatorname{sign}\left(\sum_{i=0}^{m} c^i 2^{l(m-i)}\right)$ but the cost turns out to be prohibitive as the integers involved are large and only one bit is gained at each step.

Instead we try to decrease the complexity of the computation by using intervals for the coefficients, where the bounds of the intervals are derived from the coefficients by truncating and rounding. Writing the expansion of Q_n near the target root u_i:

$$Q_n(u) = (u - u_i)Q'_n(u_i) + \mathcal{O}((u - u_i)^2)$$

so it is expected that the required precision used to compute $Q_n(u)$ and to successfully tell the sign despite the roundoff errors is approximately the number of current known good bits $-\log_2 |u - u_i|$, up to a constant factor. If the interval computed for $Q_n(u)$ contains 0, the coefficients of Q_n are truncated to a higher precision until we can decide.

The algorithm stops when we have reached the desired accuracy p in number of bits, that is $\log_2(c) > p$.

2.2.2 Newton iteration

Once the isolating interval for each root is sufficiently small, the faster Newton iteration is used. Since our error analysis requires a guaranteed error bound on the root, we use the interval Newton iteration described in [7].

2.3 Weights

The weights $(w_i)_{0 \leq i < n}$ verify

$$\int_{-1}^{1} p(x)dx = \sum_{i=0}^{n-1} w_i p(x_i)$$

for every polynomial of degree $\leq 2n-1$ (see section 3.1).
For $i \in [0, n-1]$ we write $L_i(x) = \prod_{j \neq i}(x - x_j)$. Notice that $L_i(x) = \frac{P_n(x)}{(x-x_i)P'_n(x_i)}$. L'_i has

degree $n-2$ so by definition $< L_i', P_n >= 0$.

$$0 = \int_{-1}^{1} P_n(x)L_i'(x)dx = [P_n(x)L_i(x)]_{-1}^{1} - \int_{-1}^{1} P_n'(x)L_i(x)dx.$$

$P_n'L_i$ has degree $2n-1$ so it is integrated exactly by the method:

$$0 = \frac{P_n^2(1)}{(1-x_i)P_n'(x_i)} - \frac{P_n^2(-1)}{(-1-x_i)P_n'(x_i)} - \sum_{j=0}^{n-1} w_j P_n'(x_j)L_i(x_j).$$

From Equation (1) we can see that $|P_n(\pm 1)| = 1$. Moreover $L_i(x_j) = \delta_{i,j}$ (Kronecker delta) so

$$0 = \frac{2}{(1-x_i^2)P_n'(x_i)} - w_i P_n'(x_i),$$

$$w_i = \frac{2}{(1-x_i^2)P_n'^2(x_i)}. \tag{2}$$

In Section 2.2 we showed how to compute x_i with desired accuracy. Since P_n' is known exactly we can evaluate $P_n'(x_i)$ with a dynamic error bound (known as *running error* in [8]), and from that compute w_i with arbitrary accuracy.

3 Error bounds

3.1 Mathematical error

In this section we prove the bound on the mathematical error made with the Gauss-Legendre quadrature method. A generic proof for any weight function w can be found in [5]. We adapt and recall it here for completeness. Some proofs are omitted and can be found in [9].

Theorem 1. *The Gauss-Legendre method on $[a, b]$ with n points is exact for polynomials of degree $\leq 2n-1$.*

PROOF: by definition, the Gauss-Legendre quadrature scheme being of interpolatory type, it is exact of polynomials of degree $\leq n-1$. Let f be a polynomial of degree $\leq 2n-1$. We write

$$f = q \cdot P_n + r, \quad \text{with } \deg(q) \leq n-1, \deg(r) \leq n-1.$$

Since P_n is orthogonal to the set $\mathcal{P}_{n-1}$ of polynomials of degree $\leq n-1$ we have

$$\int_{-1}^{1} q(x)P_n(x)dx = 0$$

and

$$\int_{-1}^{1} r(x)dx = I(r)$$

is computed exactly by the method. □

Let $E[f] = \int_{-1}^{1} f(x)dx - \sum_{i=0}^{n-1} w_i f(x_i)$ be the error of the method for the function f. Let h be the polynomial of degree $\leq 2n-1$ such that

$$\forall i \in [0, n-1], f(x_i) = h(x_i) \quad \text{and} \quad f'(x_i) = h'(x_i).$$

Then the remainder theorem for polynomial interpolation states that

$$f(x) = h(x) + \frac{f^{(2n)}(\zeta(x))}{(2n!)}(x-x_0)^2(x-x_1)^2\dots(x-x_{n-1})^2$$

for $-1 \leq x \leq 1$ and $a < \zeta(x) < b$. From Theorem (1), $E[h] = 0$ so we have

$$\begin{aligned} E[f] &= E\left[\frac{f^{(2n)}(\zeta(x))}{(2n!)}(x-x_0)^2(x-x_1)^2\dots(x-x_{n-1})^2\right] \\ &= -\int_{-1}^{1} \frac{f^{(2n)}(\zeta(x))}{(2n!)}\frac{P_n^2(x)}{k_n^2}dx. \end{aligned}$$

With the mean value theorem there exists $\zeta \in]-1,1[$ such that

$$E[f] = -\frac{f^{(2n)}(\zeta)}{k_n^2(2n!)}\int_{-1}^{1} P_n^2(x)dx.$$

By multiplying Equation (1) by $P_{n-1}(x)$ and integrating over $[-1,1]$ we get

$$(2n+1)\int_{-1}^{1} xP_{n-1}(x)P_n(x)dx = n\int_{-1}^{1} P_{n-1}^2(x)dx.$$

Let $v_n = \int_{-1}^{1} P_n^2(x)dx$. By Euclidean division we can write $xP_{n-1}(x) = \frac{k_{n-1}}{k_n}P_n(x) + R(x)$ where $\deg(R) < n$. So

$$\begin{aligned} \frac{(2n+1)k_{n-1}}{k_n}\int_{-1}^{1} P_n^2(x)dx &= n\int_{-1}^{1} P_{n-1}^2(x)dx \\ \frac{(2n+1)k_{n-1}}{k_n}v_n &= nv_{n-1} \\ v_n &= \left(\frac{n}{2n+1}\right)\frac{k_n}{k_{n-1}}v_{n-1}. \end{aligned}$$

Again from Equation (1) we get $k_n = \frac{2n-1}{n}k_{n-1}$, so

$$\begin{aligned} v_n &= \frac{2n-1}{2n+1}v_{n-1} \\ v_n &= \frac{2}{2n+1} \\ k_n &= \frac{(2n-1)!}{(n-1)!n!2^{n-1}}. \end{aligned}$$

Finally the error term is

$$\begin{aligned} E[f] &= f^{(2n)}(\zeta)\frac{2[(n!)(n-1)!]^2 \cdot 2^{2n-2}}{((2n-1)!)^2(2n)!(2n+1)} \\ &= \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\zeta). \end{aligned}$$

Taking into account the scaling from $[-1,1]$ to $[a,b]$ there is an additional $\left(\frac{2}{b-a}\right)^n$ factor in k_n and $\frac{b-a}{2}$ in v_n.

Theorem 2. *Let M a bound of $|f^{(2n)}|$ on $[a,b]$, then the error of the Gauss-Legendre integration of f on $[a,b]$ with infinite precision is bounded by*

$$\frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} M.$$

3.2 Roundoff errors

Algorithm 2 Gauss-Legendre integration

INPUT: $\widehat{a}, \widehat{b-a}, (\widehat{w_i}), f, (\widehat{v_i}), n$ ▷ where w_i are the weights and v_i is defined in §3.2.
OUTPUT: $\widehat{I}$.
1: **for** $i \leftarrow 0$ to $n-1$ **do**
2: $\quad t \leftarrow \circ((\widehat{b-a}) \cdot \widehat{v_i})$
3: $\quad \widehat{x_i} \leftarrow \circ(t + \widehat{a})$
4: $\quad \widehat{f_i} \leftarrow \circ(f(\widehat{x_i}))$
5: $\quad \widehat{y_i} \leftarrow \circ(\widehat{f_i} \cdot \widehat{w_i})$
6: **end for**
7: $\widehat{S} \leftarrow \mathrm{sum}(\widehat{y_i}, i = 0 \ldots n-1)$ ▷ with Demmel and Hida algorithm [10]
8: $\widehat{D} \leftarrow \circ(\widehat{b-a})/2$
9: **return** $\circ(\widehat{D}\widehat{S}) = \widehat{I}$

For the error analysis of Algorithm 2, we need a few useful lemmas concerning the "ulp[1] calculus", as well as some definitions. The floating-point numbers are represented with radix 2 (this could be generalized for any radix but radix 2 is simpler and is natural on computers). For this section, p is the working precision, and we assume all floating-point numbers are normalized, which means in our notations that the exponent range is unbounded.

Definition 1 (Exponent, Ulp). *For a non-zero real number x we define $E(x) := 1 + \lfloor \log_2 |x| \rfloor$, such that $2^{E(x)-1} \leq |x| < 2^{E(x)}$, and $\mathrm{ulp}(x) := 2^{E(x)-p}$.*

[1] unit in the last place

For a real $x \neq 0$ and a working precision p we always have $2^{p-1}\mathrm{ulp}(x) \leq |x| < 2^p\mathrm{ulp}(x)$. If x is a floating-point number, then $\mathrm{ulp}(x)$ is the weight of the least significant bit — zero or not — in the p-bit mantissa of x. For all real x, $\mathrm{ulp}(x)$ is always greater than zero by definition.

Lemma 1. *If $c \neq 0$ and $x \neq 0$ then $c \cdot \mathrm{ulp}(x) < 2 \cdot \mathrm{ulp}(cx)$.*

Lemma 2. *Assuming no underflow (flush to zero) occurs then in all rounding modes for a non zero real x we have: $\mathrm{ulp}(x) \leq \mathrm{ulp}(\circ(x))$, where $\circ(x)$ is the rounding of x in the chosen mode with an unbounded exponent range.*

Lemma 3. *Let x a non-zero real and $\circ(x)$ its rounding to nearest on p bits. Then $|x| \leq (1+2^{-p})|\circ(x)|$.*

Lemma 4. *Let a and b be two non-zero floating-point numbers of the same sign and precision p then in all rounding modes*

$$\mathrm{ulp}(a) + \mathrm{ulp}(b) \leq \frac{3}{2}\mathrm{ulp}(\circ(a+b)).$$

Lemma 5. *For x and y real numbers and using rounding to nearest in precision p we have*

$$|\circ(\circ(x) \circ (y)) - xy| \leq \left(\frac{5}{2} + 2^{-p}\right)\mathrm{ulp}(\circ(\circ(x) \circ (y))).$$

PROOF: let $u = \circ(x)$, $v = \circ(y)$ and $z = \circ(uv)$. We can write

$$x = u(1+\theta),\ y = v(1+\theta'),\ uv = z(1+\theta'') \quad \text{where} \quad |\theta|, |\theta'|, |\theta''| \leq 2^{-p},$$

$$\begin{aligned}
|z - xy| &\leq \frac{1}{2}\mathrm{ulp}(z) + |uv - xy| \\
&\leq \frac{1}{2}\mathrm{ulp}(z) + |uv|(1 - (1+\theta)(1+\theta')) \\
&\leq \frac{1}{2}\mathrm{ulp}(z) + |uv|(2^{1-p} + 2^{-2p}) \\
&\leq \frac{1}{2}\mathrm{ulp}(z) + |z|(2^{1-p} + 2^{-2p})(1 + 2^{-p}).
\end{aligned}$$

With $|z| \leq (2^p - 1)\mathrm{ulp}(z)$ we have $|z - xy| \leq \frac{1}{2}\mathrm{ulp}(z) + (2 + 2^{-p} - 2^{1-p} - 2^{-3p})\mathrm{ulp}(z)$. □

For this section we denote by $\widehat{x}$ the value actually computed (i.e. with all roundoff errors) for a given "exact" value x, as would be computed with an infinite precision from the beginning of the algorithm. In order to provide an error bound on the numerical result given by the Gauss-Legendre method, we need to have a step-by-step look into Algorithm 2.

In addition to the parameters of Algorithm 2 we need an upper bound M of $|f^{(2n)}|$ on $[a, b]$; p is the working precision expressed in the number of bits of the mantissa, a and b given as oracles which returns their values as a rounded to nearest floating-point number of desired precision; m an upper bound of $|f'|$ on $[a, b]$. In the rest of this section we will prove our main theorem:

Theorem 3. *Let* $\delta_{\widehat{y_i}} = (\frac{5}{2} + 2^{-p})\mathrm{ulp}(\widehat{y_i}) + \left(\frac{17}{4} + 23 \cdot 2^{-p-2} + 3 \cdot 2^{-2p-1}\right) m\widehat{w_i}\mathrm{ulp}(\widehat{x_i})$*, where* $\widehat{y_i}$*,* $\widehat{w_i}$ *and* $\widehat{x_i}$ *are defined in Algorithm 2. When computing the numerical quadrature of* f *using Algorithm 2 the total error on the result is bounded by:*

$$E_{total} = \left(\frac{9}{2} + 3 \cdot 2^{-p}\right) \mathrm{ulp}(\widehat{I}) + n(1 + 2^{-p})\widehat{D} \cdot \max(\delta_{\widehat{y_i}}) + \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} M.$$

We split the error bound in three terms:

1. *the mathematical error* $\frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} M$*,*
2. *the "static" error* $E_{\mathrm{stat}} = (\frac{9}{2} + 3 \cdot 2^{-p})\mathrm{ulp}(\widehat{I})$ *which comes mostly from the rounding error in the summation,*
3. *the evaluation error* $E_{\mathrm{eval}} = n(1 + 2^{-p})\widehat{D} \cdot \max(\delta_{\widehat{y_i}})$ *which accounts for the rest of the rounding errors and tracks possible cancellations in the summation as well.*

Corollary 1. *If we assume that* $p \geq 2$ *we have the simpler formulas:*

$$\begin{cases} \delta_{\widehat{y_i}} &= \frac{11}{4}\mathrm{ulp}(\widehat{y_i}) + 6m\widehat{w_i}\mathrm{ulp}(\widehat{x_i}) \\ E_{total} &= \frac{21}{4}\mathrm{ulp}(\widehat{I}) + \frac{5n}{4}\widehat{D} \cdot \max(\delta_{\widehat{y_i}}) + \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} M. \end{cases}$$

The algorithm can be analyzed in several steps:

1. the computation of the weights w_i, $i \in [0, n-1]$ of the method. They are computed as explained in Section 2 as the rounded to nearest of the exact value:

$$|\widehat{w_i} - w_i| \leq \frac{1}{2}\mathrm{ulp}(\widehat{w_i}).$$

2. the computation of x_i. Let x'_i be the corresponding evaluation point on $[-1, 1]$, and $v_i = \frac{1+x'_i}{2}$. We assume v_i is computed as the rounded to nearest of the exact value:

$$|\widehat{v_i} - v_i| \leq \frac{1}{2}\mathrm{ulp}(\widehat{v_i}),$$

$$\widehat{x_i} = \circ(\circ(\widehat{v_i} \cdot \widehat{(b-a)}) + \widehat{a}).$$

Since a and b are given as oracles, we can assume that $b-a$ as well as a were computed as rounded to nearest of the correct value. The error analysis gives:

$$\begin{aligned} |\circ(\widehat{v_i}\cdot\widehat{b-a}) - v_i\cdot(b-a)| &\leq \left[\tfrac{5}{2}+2^{-p}\right]\mathrm{ulp}(\circ(\widehat{v_i}\cdot\widehat{b-a})) \quad \text{[Lemma 5]} \\ |\widehat{x_i}-x_i| &\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{x_i}) + \left[\tfrac{5}{2}+2^{-p}\right]\mathrm{ulp}(\circ(\widehat{v_i}\cdot\widehat{b-a})) + \tfrac{1}{2}\mathrm{ulp}(\widehat{a}) \\ &\leq (\tfrac{17}{4}+3\cdot 2^{-p-1})\mathrm{ulp}(\widehat{x_i}). \quad \text{[Lemma 4]} \end{aligned}$$

3. the computation of $f(x_i)$. We assume we have an implementation of f with correct rounding, and we call f which returns the rounding to nearest of the exact value with precision p. Such correctly rounded implementations of mathematical functions with arbitrary precision on the result can be found for example in MPFR [11] for non-trivial functions like $\exp$, $\sin$, $\arctan$ and numerous others.

   With the already estimated error on $\widehat{x_i}$ we have:

$$|f(\widehat{x_i}) - f(x_i)| = |f'(\theta_i)(\widehat{x_i}-x_i)|, \quad \theta_i \in [\min(x_i,\widehat{x_i}), \max(x_i,\widehat{x_i})]$$

   and with an upper bound on f' we can bound this error absolutely. Let $\widehat{f_i} = \circ(f(\widehat{x_i}))$ be the floating-point number computed. At this step we now have:

$$\begin{aligned} \delta_{\widehat{f_i}} = |\widehat{f_i} - f(x_i)| &\leq |f'(\theta_i)(\widehat{x_i}-x_i)| + \tfrac{1}{2}\mathrm{ulp}(\widehat{f_i}) \\ &\leq (\tfrac{17}{4}+3\cdot 2^{-p-1})m\cdot\mathrm{ulp}(\widehat{x_i}) + \tfrac{1}{2}\mathrm{ulp}(\widehat{f_i}). \end{aligned}$$

4. computation of the $y_i = f(x_i)\cdot w_i$. The accumulated error so far:

$$\begin{aligned} |\widehat{y_i}-y_i| &\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{y_i}) + |\widehat{f_i}\widehat{w_i} - f(x_i)w_i| \\ &\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{y_i}) + \widehat{f_i}|\widehat{w_i}-w_i| + w_i|\widehat{f_i}-f(x_i)| \\ &\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{y_i}) + \tfrac{1}{2}\widehat{f_i}\mathrm{ulp}(\widehat{w_i}) + w_i\delta_{\widehat{f_i}} \\ &\leq \tfrac{3}{2}\mathrm{ulp}(\widehat{y_i}) + w_i\left[(\tfrac{17}{4}+3\cdot 2^{-p-1})m\cdot\mathrm{ulp}(\widehat{x_i}) + \tfrac{1}{2}\mathrm{ulp}(\widehat{f_i})\right] \quad \text{[Lemmas 1 and 2]} \\ &\leq \tfrac{3}{2}\mathrm{ulp}(\widehat{y_i}) + (1+2^{-p}|\widehat{w_i}\left[(\tfrac{17}{4}+3\cdot 2^{-p-1})m\cdot\mathrm{ulp}(\widehat{x_i}) + \tfrac{1}{2}\mathrm{ulp}(\widehat{f_i})\right] \quad \text{[Lemma 3]} \\ &\leq (\tfrac{5}{2}+2^{-p})\mathrm{ulp}(\widehat{y_i}) + (1+2^{-p})m\widehat{w_i}(\tfrac{17}{4}+3\cdot 2^{-p-1})\mathrm{ulp}(\widehat{x_i}) \quad \text{[Lemmas 1 and 2]} \\ &\leq (\tfrac{5}{2}+2^{-p})\mathrm{ulp}(\widehat{y_i}) + \left(\tfrac{17}{4}+23\cdot 2^{-p-2}+3\cdot 2^{-2p-1}\right)m\widehat{w_i}\mathrm{ulp}(\widehat{x_i}) = \delta_{\widehat{y_i}}. \end{aligned}$$

   Remark: when bounding the error on $\widehat{x_i}$, $\widehat{f_i}$ as well as $\widehat{y_i}$, the term with $\mathrm{ulp}(\widehat{x_i})$ vanishes if the error on $\widehat{x_i}$ is zero. One can easily show with our assumption that no underflow occurs, and that if $\widehat{x_i}=0$ then the error on $\widehat{x_i}$ is zero (i.e. $x_i=0$) and the ill-defined quantity $\mathrm{ulp}(\widehat{x_i})$ vanishes. For the error bound we keep track of only $\max(\delta_{\widehat{y_i}})$.

5. summation of the y_i's: this is done with Demmel and Hida summation algorithm [10], which guarantees an error of at most 1.5 ulp on the final result. This algorithm uses a larger working precision $p' \approx p + \log_2(n)$. Let $S = \sum_{i=0}^{n-1} y_i$.

$$|\widehat{S} - S| \leq \frac{3}{2}\mathrm{ulp}(\widehat{S}) + n\cdot\max(\delta_{\widehat{y_i}}).$$

6. multiplication by $\frac{b-a}{2}$: $I = \frac{b-a}{2}S$. We note $D = \frac{b-a}{2}$ and assume as before that the input $\widehat{b-a}$ was computed as the rounded to nearest of its exact value. Since the division by 2 is exact we have:

$$\begin{aligned}
|\widehat{D} - D| &\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{D}) \\
|\widehat{I} - I| &\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{I}) + |\widehat{S}\widehat{D} - SD| \\
&\leq \tfrac{1}{2}\mathrm{ulp}(\widehat{I}) + \tfrac{1}{2}|\widehat{S}|\mathrm{ulp}(\widehat{D}) + D|\widehat{S} - S| \\
&\leq \tfrac{3}{2}\mathrm{ulp}(\widehat{I}) + D\left[\tfrac{3}{2}\mathrm{ulp}(\widehat{S}) + n\cdot\max(\delta_{\widehat{y}_i})\right] \quad \text{[Lemmas 1 and 2]} \\
&\leq \tfrac{3}{2}\mathrm{ulp}(\widehat{I}) + (1+2^{-p})\widehat{D}\left[\tfrac{3}{2}\mathrm{ulp}(\widehat{S}) + n\cdot\max(\delta_{\widehat{y}_i})\right] \quad \text{[Lemma 3]} \\
&\leq (\tfrac{9}{2} + 3\cdot 2^{-p})\mathrm{ulp}(\widehat{I}) + n(1+2^{-p})\widehat{D}\cdot\max(\delta_{\widehat{y}_i}). \quad \text{[Lemmas 1 and 2]}
\end{aligned}$$

Corollary 2. *If we assume furthermore that f does not change sign on $[a,b]$, then we have the following better bound:*

$$\begin{aligned}
E_{total} = &\left(\frac{9}{2} + 3\cdot 2^{-p} \quad + (2+2^{1-p})(5+2^{1-p})(1+3\cdot 2^{-p})\right)\mathrm{ulp}(\widehat{I}) \\
&+ nm(1+2^{-p})\left(\frac{17}{4} + 23\cdot 2^{-p-2} + 3\cdot 2^{-2p-1}\right)\widehat{D}\max(\widehat{w_i}\mathrm{ulp}(\widehat{x_i})) \\
&+ \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3}M.
\end{aligned}$$

PROOF: Let us assume for example that $f \geq 0$, then we have

$$\forall i \in [0, n-1], \widehat{y}_i = \circ(\widehat{w_i}\cdot\widehat{f_i}) \geq 0$$

so

$$\mathrm{ulp}(\widehat{y_i}) \leq 2^{1-p}\widehat{y_i}.$$

Let $\tilde{S} = \sum_{i=0}^{n-1}\widehat{y_i}$, we know that

$$\begin{aligned}
|\tilde{S} - \widehat{S}| &\leq \tfrac{3}{2}\mathrm{ulp}(\widehat{S}) \\
\tilde{S} &\leq (1+3\cdot 2^{-p})\widehat{S} \\
L &= \textstyle\sum_{i=0}^{n-1}(\frac{5}{2} + 2^{-p})\mathrm{ulp}(\widehat{y_i}) \\
&\leq (\tfrac{5}{2} + 2^{-p})2^{1-p}\textstyle\sum_{i=0}^{n-1}\widehat{y_i} \\
&\leq 2^{1-p}(\tfrac{5}{2} + 2^{-p})(1+3\cdot 2^{-p})\widehat{S} \\
&\leq (5+2^{1-p})(1+3\cdot 2^{-p})\mathrm{ulp}(\widehat{S})
\end{aligned}$$

From this we get the following bound on the error on $\widehat{S}$:

$$|\widehat{S} - S| \leq \left(\frac{3}{2} + (5+2^{1-p})(1+3\cdot 2^{-p})\right)\mathrm{ulp}(\widehat{S}) + nm\left(\frac{17}{4} + 23\cdot 2^{-p-2} + 3\cdot 2^{-2p-1}\right)\max(\widehat{w_i}\mathrm{ulp}(\widehat{x_i}))$$

and substituting this expression in the bound of $|\widehat{I} - I|$ above yields the announced result.

Since $0 \leq \widehat{w_i} \leq 1$ and assuming $p \geq 2$ we can write the simpler bound

$$E_{total} = 30\mathrm{ulp}(\widehat{I}) + 8nm\widehat{D}\max(\mathrm{ulp}(a), \mathrm{ulp}(b)) + \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3}M.$$

□

4 Experiments

Algorithm 2 was implemented using the MPFR library [11]. In addition to the result of the integration, the program gives an error bound E_{total} on the computed result split in three terms, as explained in Theorem 3. For our experiments we chose a function and an

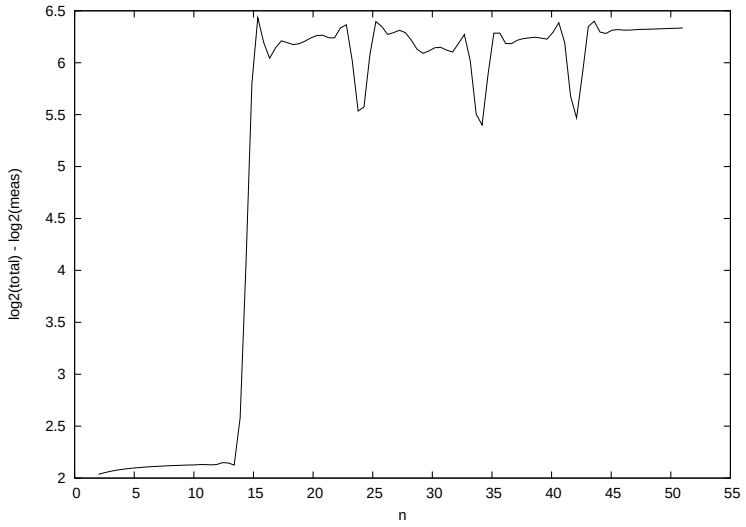

Figure 1: The overestimation of the error in bits when computing $\int_0^3 e^x dx$ with 113 bits of precision and n evaluation points.

integration domain where the exact value is known, so that we can measure precisely the actual error of the computation (denoted by E_{meas}). Figure 1 shows how much pessimistic the error bound is in number of bits, computed as $\log_2(E_{\mathrm{total}}) - \log_2(E_{\mathrm{meas}})$ when computing the integral $I = \int_0^3 e^x dx$ with 113 bits of working precision, the number of evaluation points varying from 2 to 100.

The error due to roundoffs ($E_{\mathrm{stat}} + E_{\mathrm{eval}}$) is quite stable. This is expected of the Gauss-Legendre integration scheme where the weights are all positive. With this property the

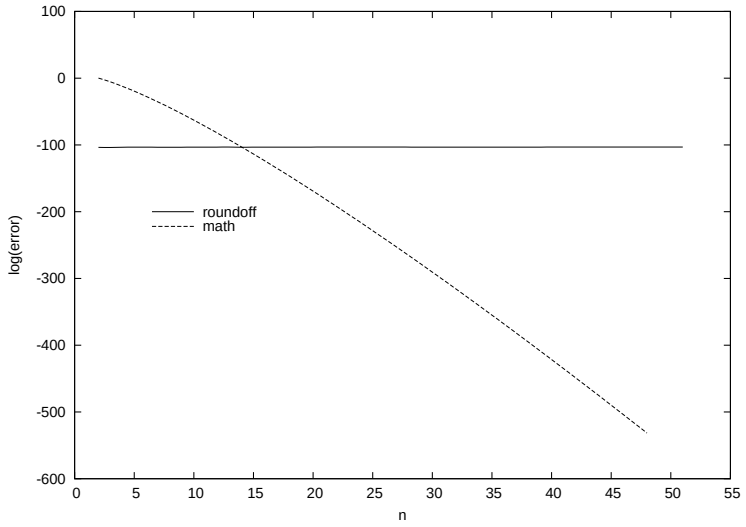

Figure 2: The roundoff errors $E_{\mathrm{eval}} + E_{\mathrm{stat}}$ and the mathematical error when computing $\int_0^3 e^x dx$ with 113 bits of precision and n evaluation points.

integration can benefit from an increase in the number of evaluation points n in order to compute a better result. This is not the case for the Newton-Cotes method for example.

The bound on the total error as given by the algorithm is close to the measured error. In the experimental data we observe a maximum loss of 7 bits of precision on the computed result due to a pessimistic bound. However, this overestimation of the error is noticable for $n \geq 15$ which is the point where the mathematical error becomes less than the roundoff errors. Quite naturally our overestimation is made on the roundoff errors and not on the mathematical error.

5 Conclusion

The Gauss-Legendre quadrature scheme provides a robust numerical integration algorithm. In this paper we performed an error analysis of the algorithm that can be used to determine dynamically an error bound on the final result. From this we can easily derive a correctly rounded numerical quadrature algorithm, provided the result is not exactly representable by a floating-point number.

The running time of the algorithm is dominated by the cost of computing the evaluation points and weights (Figure 3). As future work it is planned to improve this time, for example by using precomputed tables of roots optimized for the space.

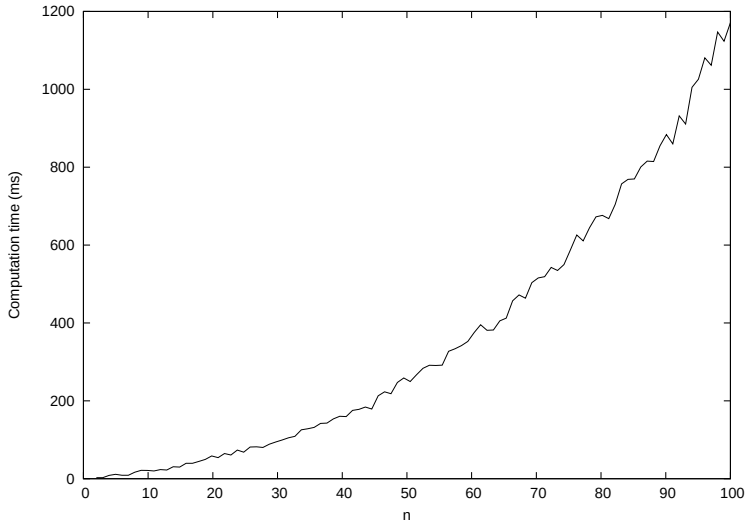

Figure 3: Roots and weights computation time for 113 bits of precision and n evaluation points. Measurements made on a 3GHz Pentium 4 processor.

p	n_{opt}	Predicted good bits
53	8	47
113	15	108
200	22	194
400	38	395
1000	80	995

Figure 4: Optimized order for different working precisions p in bits.

Another idea to work on is to consider composition of the integration method and chose automatically the best composition level and n for a given function, integration domain and the required precision with emphasis on the running time. Figure 4 gives the smallest n for which the mathematical error is smaller than the computed bound on the roundoff error $E_{\mathrm{eval}} + E_{\mathrm{stat}}$ in the computation of $\int_0^3 \exp(x)dx$, and gives a first hint in this direction.

References

[1] IEEE standard for binary floating-point arithmetic. Tech. Rep. ANSI-IEEE Standard 754-1985, New York, 1985. approved March 21, 1985: IEEE Standards Board, approved

July 26, 1985: American National Standards Institute, 18 pages.

[2] WALTER OEVEL Numerical Computations in MuPAD 1.4 In *mathPAD vol 8 No 1*, 1998.

[3] F. JEZEQUEL, M. CHARIKHI, J.-M. CHESNEAUX Dynamical control of computations of multiple integrals, SCAN2002 conference, Paris (France) 23-27 September 2002.

[4] DAVID H. BAILEY, XIAOYE S. LI A Comparison of Three High-Precision Quadrature Schemes In Proceedings of Real Numbers And Computers (RNC5), Lyon (France) 3-5 September 2003.

[5] PHILIP J. DAVIS AND PHILIP RABINOWITZ Methods of numerical integration. 1984

[6] FABRICE ROUILLIER AND PAUL ZIMMERMANN Efficient isolation of a polynomial real roots INRIA Research report 4113, `http://www.inria.fr/rrrt/rr-4113.html` February 2001.

[7] NATHALIE REVOL Interval Newton iteration in multiple precision for the univariate case École Normale Supérieure de Lyon and INRIA Research report RR-4334 `http://www.inria.fr/rrrt/rr-4334.html`

[8] NICHOLAS J. HIGHAM Accuracy and Stability of Numerical Algorithms, second edition SIAM 2002.

[9] LAURENT FOUSSE Correctly rounded Newton-Cotes Quadrature, INRIA Research report (submitted), `http://www.inria.fr/rrrt/rr-5605.html` 2005.

[10] DEMMEL, J., AND HIDA, Y. Accurate floating point summation. `http://www.cs.berkeley.edu/~demmel/AccurateSummation.ps`, May 2002.

[11] THE SPACES PROJECT. The MPFR library, version 2.1.2. `http://www.mpfr.org/`, 2005.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399